Fibonacci Sequence Generation with Stream Programming Paradigm

Saeid Yousefpour*[1], Yalda Sani[2]

Department of Computer Science, Islamic Azad University, Shabestar Branch, Shabestar, Iran

*[1]saeid.yousefpour@gmail.com; [2]sani.yalda@gmail.com

Abstract

Fibonacci sequence is one of the important problems in mathematics and real life. Also it is widely applied in computer science. There are several computer algorithms to generate this sequence. We have used stream programming paradigm to generate Fibonacci sequence in java. For this reason we have developed a java library that includes stream programming models based on StreamIT language. Our codes are simple java code and there is no need to learn any new syntax. Also in our code there is not any recursive function call, in order to prevent increasing time complexity of the algorithm.

Keywords

Fibonacci Sequence; Stream Programming; StreamIT Language

Introduction

Leonardo Fibonacci was born in Pisa, Italy, and was the most skilled mathematician of the middle ages. In the 12th century, he was doing a research on population growth rate of rabbits, under ideal circumstances, such as no predator to eat them, no lack of food or water and anything else that would affect the growth rate. The Fibonacci numbers are generated as a result of solving this problem as shown in fig1.

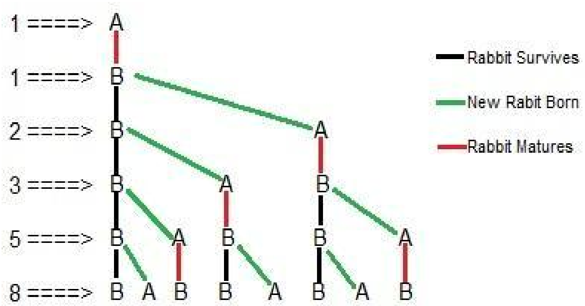

FIG. 1 FIBONACCI SEQUENCE IN RABBITS POPULATION

The first two numbers in the Fibonacci sequence are 0 and 1 (alternatively, 1 and 1), and each subsequent number is the sum of the previous two. So firstly take "1" and add it to the previous number "Zero", the result is "1", then add "1" to" 1" the result is "2" and use the same way to get the rest numbers of the sequence.

So the Fibonacci sequence is a recursive sequence defined by the equations below:

$$F_1 = 1, \ F_2 = 1 \quad and \quad F_n = F_{n-1} + F_{n-2} \quad for\ all \quad n \geq 3$$

Where F_n represents the n_{th} Fibonacci number (n is called index).

In computer science there are many algorithms to produce Fibonacci sequence. Each of these algorithms uses different methods to generate the sequence. Nevertheless they can be classified into three main categories. Algorithms of the first category describe the Fibonacci function with a Fibonacci relation as given below:

$$F(0) = 1$$
$$F(1) = 1$$
$$for\ all\ n > 1\ F(n) = F(n-1) + F(n-2)$$

A recursive function, base on the relation that mentioned above, takes exponential arithmetic operation. In the second category, algorithms use a table to store partial results in the recurrence relation computation and therefore results are in linear arithmetic operations. The third category contains the algorithms which are able to compute the function in logarithmic time.

In this paper we have used stream programming paradigm to generate Fibonacci sequence. A stream program is a type of computer program in which data are stream of data. The "stream of data" means that data are logically infinite and continually come from a source into the program. For example imagine a signal processing system. In a regular signal processing system, signals come from an external resource into program and functions of the program execute on the signal. These input signals are streams of data. Speech

encoding and image processing systems are other examples of stream processing systems.

Stream programming, also is an applicable method to write parallel applications on multi-core architecture. Multi-core processors deliver high performance through multi-processor parallel processing. Continued use of sequential programs on multi-core computer is a waste of multiprocessor resources so converting sequential programs to parallel programs is an urgent. The stream programming paradigm is a software approach to writing parallel applications on multi-core architecture.

In this paper we have used JStream java library to generate Fibonacci sequence in streaming model. The library has some base classes for computational models, but in this paper we describe two of them that are used in Fibonacci sequence.

The rest of this paper is organized as follow: Section2 is a description of some base classes which are used in Fibonacci sequence. Section 3 presents the structure and the source code of Fibonacci sequence in streaming model and at the end section 4 is conclusion.

Base Classes Used in Fibonacci Sequence

In order to develop stream programs in java, we have developed JStream library. The library has some base classes for filter and data communication patterns. Our library is based on StreamIT language. In StreamIT each functional unit is called Filter. Also it has three patterns to arrange filters for data communications through communication channels. Each filter has one input channel and one output channel. It reads data from input channel, processes data and writes the result to output channel.

In addition there are three communication patterns in StreamIT: pipeline, feedback loop and split-join. Filters and patterns generally are called streams. In Fibonacci sequence we have used pipeline and feedback loop. At the rest of this section we will briefly explain these two patterns and filter structure.

Filter

The JStream library has a base class for filters. If a programmer wants to create a filter, he can simply inherit from Filter class of the library. Each filter has three operations on its input and output channels. These operations are pop(), peek() and push(). pop and peek are input operations and push is an output operation. pop() removes data from input channel but peek() just reads the value of data from input channel. push() writes results into output channel. The filter definition syntax is shown below:

public class FilterName extends

Filter<InputType,OutputType>

Each filter should contain a function by the name of run (). This function is the main function in the filter and includes the code that filter should execute when running.

Pipeline

Pipeline is the simplest form of communication patterns. A pipeline has a number of child streams and the output of first stream is connected to the input of second stream and the output of second stream is connected to the input of third stream and so on. Fig. 2 shows a pipeline structure.

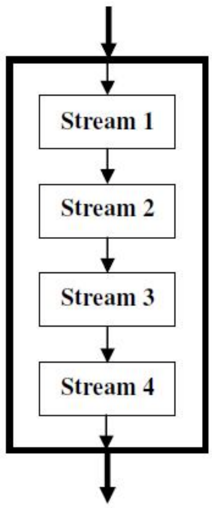

FIG. 2 PIPELINE STRUCTURE

We are not allowed to define any extra function in the pipeline class. Input data type of pipeline should be the same as the input data type of first child. Also the utput type of pipeline should be the same as the output type of the last child. The following syntax shows pipeline definition.

public class PipeLineName extends

PipeLine<InputType,OutputType>

Pipeline has a method named Add(). It is used to add a child stream into pipeline.

Feedback Loop

A feedback loop pattern has a body stream. Output of the body stream is sent to a splitter. One branch of splitter leaves the loop and another branch is returned back to the body through a joiner. The joiner joins the input channel of feedback loop to the loop channel. Data type of input channel, output channel and loop branch should be the same. Fig. 3 shows a feedback loop.

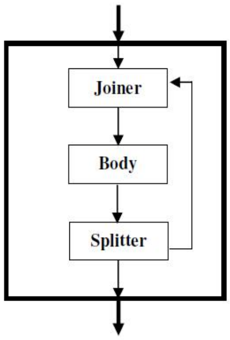

FIG. 3 FEEDBACK LOOP STRUCTURE

The following lines show feedback loop definition syntax.

```
public class FeedBackLoopName extends
FeedBackLoop<InputType,OutputType>
```

Generating Fibonacci Sequence

In this section we will explain the way that Fibonacci sequence generated with our library. For this reason we have created a filter and have named it FiboFilter that computes Fibonacci equation. The source code below shows FiboFilter.

```
public class FiboFilter extends Filter <Integer,Integer>
 {
   public static FiboFilter FibConstruct()
    {
        FiboFilter Obj=new FiboFilter();
        return Obj;
    }
   public void run()
    {
        Push(Peek(1)+Pop());
    }
 }
```

The class FiboFilter extends filter class, so it is a filter. As mentioned above, each filter should have a run() method. The run() method adds two data items of input channel and pushes the result into output channel. But the first item is removed after addition and the second item remains in channel for generating the next number.

Now we should import the generated number into input channel of FiboFilter. To do this we have to use feedback loop structure. So the FiboFilter should be replaced as the body of feedback loop. The following source code shows how to create a feedback loop with FiboFilter.

```
 public class FiboFeedBackLoop extends FeedBackLoop
 <Integer,Integer>
 {
 public FiboFeedBackLoop()
   {
     SetBody(FiboFilter.FibConstruct());
     SetJoiner(0,1);
     SetSplitter(1,1);
     Enqueue(0);
     Enqueue(1);
   }
 public static FiboFeedBackLoop FibLoopConstruct()
{
 FiboFeedBackLoop Obj=new  FiboFeedBackLoop();
        return Obj;
  }
}
```

FiboFeedBackLoop is a class that extends FeedBackLoop base class. So it becomes a feedback loop structure. Constructor method for this class includes some functions. We give an explanation for each one.

SetBody(Stream Name) identifies the body stream of feedback loop. Here FiboFilter is the body and Feedback loop is a stream. So it has its own input and output channels as well as a loop channel (see fig. 3). Feedback loop has a joiner that joins input channel data to loop channel data. Here SetJoiner(x,y) means push *x* number of data items from input channel and y

number of data items from loop channel into body input channel. Also SetJoiner(0,1) means don't get any data from input channel and get one data from loop channel in every iteration. SetSplitter(1,1) means push one copy of data to output channel and one copy to loop channel (see position of splitter in fig. 3).

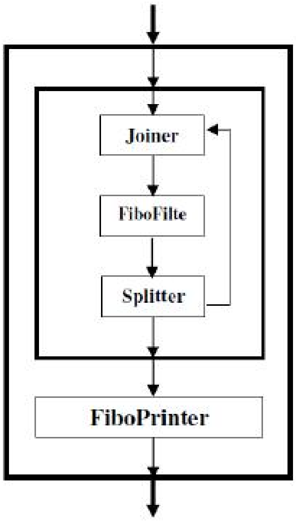

FIG. 4 FIBONACCI SEQUENCE STREAM GRAPH

Enqueue(x) inserts data *x* into input channel of body stream, So Enqueue(0) and Enqueue(1) insert 0,1 into input channel of FiboFilter. In Fibonacci sequence two initial values are 0,1. Fig. 4 illustrates Fibonacci equence stream graph. FiboPrinter is a simple filter that prints input channel data to standard output.

Conclusion

The Fibonacci sequence is one of the important problems in mathematics, computer science and real life. There are lots of algorithms to generate this sequence by computer. Some of these algorithms are recursive algorithms and some use tables to store intermediate results. In this paper we have introduced a new way for generating the Fibonacci sequence by using Stream Programming paradigm.

We developed a java library that includes stream programming processing model based on StreamIT language. Our source code is a simple java code and there is no recursive function calling in the code.

REFERENCES

Arvind, “Data flow languages and architectures”, Poceedings of the 8th annual symposium on computer architecture, USA, 1981.

Das, Abhishek, Dally, William J., Mattson, Peter, “Compiling for stream processing”, proceedings of the 15th international conference on parallel architectures and compilation techniques, Seattle, Washington, USA, 2006.

Donoghue, John, "State estimation and control of the Fibonacci system", Journal Signal Processing, Volume 91 Issue 5, P 1190-1193, 2011.

Gordon, Michel I., Thies, William, Amarasinghe, Saman, “Exploiting coarse-grained task, data, and pipeline parallelism in stream programs”, proceedings of the 12th international conference on architectural support for programming languages and operating systems, San Jose, California, USA, 2006.

Johnson, L.F., "Tumble, a fast simple iteration algorithm for Fibonacci", Journal Information processing letters, vol 89, Issue 4, p 187-189, 2004.

Karan, Mathu Manikandan Bas, Vydyanathan, Nagavijayalakshmi, Bondhugula, Uday Kumar Reddy, Ramanujam, J., “Compiler assisted dynamic scheduling for effective parallelization of loop nested on multi core processors”, proceedings of the 14th ACM SIGPLAN synposium on principles and practice of parallel programming, Raleigh, USA, 2009.

Karczmarek, Michel, Thies, William, Amarasinghe, Saman, “Phased schrduling of Stream prgrams”, proceedings of the 2003 ACM SIGPLAN conference on language. Compiler and tool for embeded systems, San Diego, California, USA, 2003.

Lamb, Andrew A., Thies, William, Amarasinghe, Saman, “Linear analysis and optimization of stream programs”, San Diego, California, USA, 2003.

Lio, Shin-Wei, Du, Zhaohui, Wu, Ganasha, Lueh, Guei-Yuan, “Data and computation transformation for brook streaming applications on multi processors”, proceedings of the international symposium on code generation and optimization, 2006.

Liu, Duo, Shao, Zili, Wang, Meng, Guo, Minyi, Xue, Jingling, “Optimal loop parallelization for maximizing iteration-level parallelism”, proceedings of the 2009 international conference on compilers, architecture, and synthesis for embeded systems, Grenoble, France, 2009.

Liu, Lixia, Li, Zhiyuan,, “Improving parallelism and locality with asynchronous algorithms”, proceedings of the 10th ACM SIGPLAN symposium on principles and practicee

of parallel programming, Bangalore, India, 2010.

Spring, JasperH., Privat, Jean, Guerraoui, Rechid and Vitek, Jan, “StreamFlex: High-Throughput Stream Programming in Java”, Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications, OOPSLA'07, Montreal, Canada, 2007.

“StreamIt: a language for streaming application”, online resource: http://csail.mit.edu.

”StreamIt language specification”, online resource; http://csail.mit.edu.

Thies, William, “Language and compile support for stream programs”, PhD theses, MIT, 2009.

Waingold, Elliot L., “SIFt: A Compiler for Streaming Applications”, MSc Theses, MIT, 2000.

Yang, Xuejun, Zhang, Ying, Xue, Jingling, Rogers, Ian, Li, Gen, Wang, Guibin, “Exploiting loop-dependent stream reuse for stream processors”, proceedings of the 17th international conference on parallel architectures and compilation techniques, New York, USA, 2008.